DAM
Desarrollo de Aplicaciones Multiplataforma
2º Curso

AD
Acceso a Datos

UD 4
Modelo Objeto Relacional (ORM)

IES BALMIS
Dpto Informática
Curso 2019-2020
Versión 1 (03/2019)

UD4 – Modelo Objeto Relacional (ORM)

ÍNDICE

9. Anotaciones en Hibernate

El formato XML tiene muchos detractores, que lo consideran "**demasiado farragoso**" y poco legible en documentos de un cierto tamaño. En ciertos entornos se prefieren notaciones más compactas, como JSON. En Java, en ciertos casos se pueden emplear "**anotaciones**", etiquetas especiales precedidas por el símbolo de la arroba (@).

Hibernate permite también usar anotaciones en vez de ficheros XML.

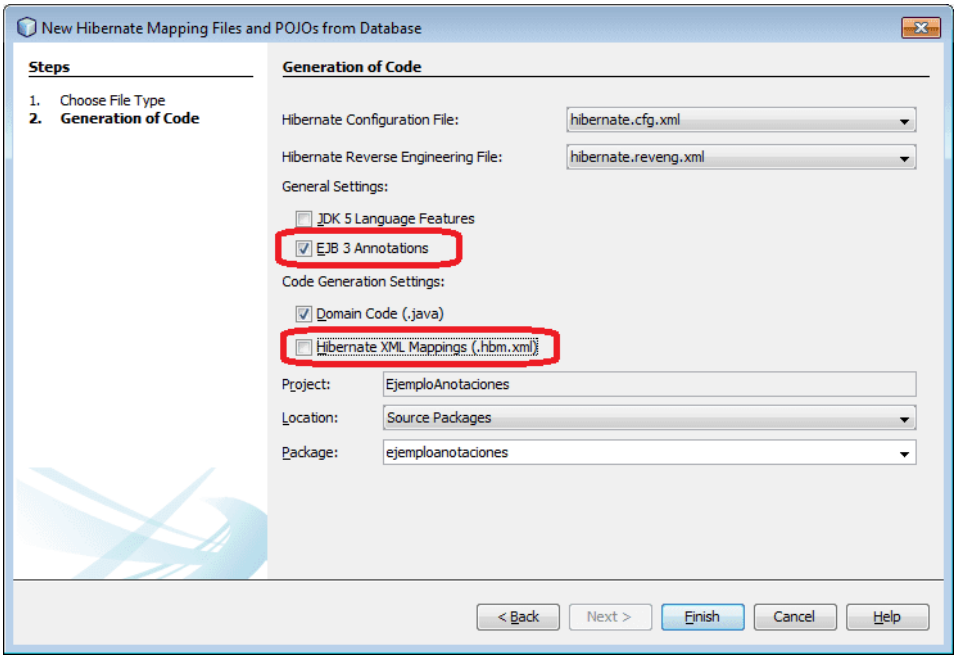
Para ver un ejemplo, vamos a crear una nueva base de datos **bdanotaciones** que tenga solo una tabla simple, con libros, aprovechando parte de la estructura que ya habíamos creado antes.

```
CREATE TABLE libros (
  id        INTEGER PRIMARY KEY,
  titulo    VARCHAR(60)
);

INSERT INTO libros ( id, titulo) VALUES
  (1, 'Macbeth'),
  (2, 'La Celestina (Tragicomedia de Calisto y Melibea)'),
  (3, 'El Lazarillo de Tormes'),
  (4, '20.000 Leguas de Viaje Submarino'),
  (5, 'Alicia en el País de las Maravillas'),
  (6, 'Cien Años de Soledad');
```

Ahora, deberemos crear un nuevo proyecto (**EjemploAnotaciones**, por ejemplo), crear la configuración básica de Hibernate, luego el fichero de ingeniería inversa, y después escoger que queremos crear los ficheros de mapeado y POJOs.

Esta vez seleccionaremos la opción de "**EJB 3 Annotations**" y desactivaremos la de "**Hibernate XML Mappings**".



El resultado será que no se creará el fichero "**libros.hbm.xml**" y que la clase Libros será con anotaciones.

```
package ejemploanotaciones;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

/** * Libros generated by hbm2java */
@Entity
@Table(name="libros" ,schema="public" )
public class Libros implements java.io.Serializable {

    private int id;
    private String titulo;

    // Constructores
    public Libros() {
    }
    public Libros(int id) {
        this.id = id;
    }
    public Libros(int id, String titulo) {
       this.id = id;
       this.titulo = titulo;
    }

    @Id
    @Column(name="id", unique=true, nullable=false)
    public int getId() {
        return this.id;
    }
    public void setId(int id) {
        this.id = id;
    }

    @Column(name="titulo", length=60)
    public String getTitulo() {
        return this.titulo;
    }
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

}
```

Y también cambiará la sección "**mapping**" del fichero de configuración básica de Hibernate, "**hibernate.cfg.xml**", para hacer referencia a esta clase, en vez de al (inexistente) fichero hbm.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/bibliotecah?
zeroDateTimeBehavior=convertToNull&amp;autoReconnect=true&amp;useSSL=false</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">1234</property>
    <property name="hibernate.show_sql">true</property>
    <property name="hibernate.query.factory_class">
       org.hibernate.hql.internal.classic.ClassicQueryTranslatorFactory</property>
    <mapping class="ejemploanotaciones.EjemploAnotaciones" />
    <mapping class="ejemploanotaciones.Libros" />
  </session-factory>
</hibernate-configuration>
```

Tras generar el HibernateUtil, todo debería funcionar igual que antes.